AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions and listings of claims in the application:

Listing of Claims:

1. **(Currently Amended)** A method of translating binary code instructions from a source format to a target format for processing by a target processor, said method comprising the steps of:

identifying a source instruction;

selecting a translation template corresponding to said identified source instruction, said template providing a set of target instructions semantically equivalent to said identified source instruction;

translating said identified instruction in accordance with said template;

~~performing dependency analysis using a Directed Acyclic Graph;~~

generating dependency analysis code using input and output resources named in the template, the generating of dependency analysis code including:

maintaining a counter associated with each instruction to indicate a group number to which the instruction belongs, wherein all instructions of a same group are issued in parallel; and

assigning each instruction to an earliest group of instructions such that all producers of input and output resources of the instruction have already been assigned to previous groups; and

outputting said translated instruction for processing by said target processor.

2. (Original) A method according to claim 1 in which said source and target instructions include a control part and a data part and said control part being used in said identification step to identify an instruction.

3. (Previously Presented) A method according to claim 2 in further comprising a transformation step in which said data part from said source instruction is transformed into said corresponding data part or parts of said set of target instructions.

4. (Original) A method according to claim 3 in which said transformation step is carried out in accordance with a bit filling routine associated with said template.

5. (Original) A method according to claim 4 in which said bit filling routine is uniquely associated with said template.

6. (Original) A method according to claim 3 in which said transformation step is arranged to transform data of one type of endianness to data of another type of endianness.

7. (Original) A method according to claim 2 in which said source instruction control parts are each concatenated to provide a unique identifier and said templates are indexed in accordance with said identifiers.

8. (Original) A method according to claim 7 in which said templates are indexed by said unique identifiers in a look up table.

9. (Original) A method according to claim 1 in which said translation is carried out at runtime of an emulated application program.

10. (Original) A method according to claim 1 in which said templates are provided by software procedure calls.

11. (Original) A method according to claim 1 in which said source format is 32 bit and said target format is 64 bit.

12. (Original) A method according to claim 1 in which said source format is PA-RISC code and said target format is Itanium™ code.

13. **(Currently Amended)** Apparatus for translating binary code instructions from a source format to a target format for processing by a target processor, the apparatus comprising:

an instruction identifier for identifying a source instruction;

a template selector for selecting a translation template corresponding to said identified source instruction, said translation template comprising a set of target instructions semantically equivalent to said identified source instruction and further comprising input and output resources; and

a translator for translating said identified instruction in accordance with said template;

a scheduler that performs dependency analysis using a ~~Directed-Acyclic Graph~~ counter associated with each instruction to indicate a group number to which the instruction belongs, wherein the counter computes the earliest group to which a given instruction can be assigned such that all producers of inputs and outputs of the instruction have already been assigned to previous groups ~~to represent dependencies~~;

an analysis routine generator that generates dependency analysis code ~~using the input and output resources named in the template~~using the dependency analysis and

an output buffer for outputting said translated instruction for processing by said target processor.

14. (Original) Apparatus according to claim 13 in which said source and target instructions include a control part and a data part and said instruction identifier uses said control part to identify an instruction.

15. (Previously Presented) Apparatus according to claim 14 in which in said translator is operable to transform said data part from said source instruction into said corresponding data part or parts of said set of target instructions.

16. (Original) Apparatus according to claim 15 in which said transformation is carried out in accordance with a bit filling routine associated with said template.

17. (Original) Apparatus according to claim 16 in which said bit filling routine is uniquely associated with said template.

18. (Original) Apparatus according to claim 15 in which translator is operable to transform data of one type of endianness into data of another type of end ianness.

19. (Original) Apparatus according to claim 14 in which said source instruction control parts are concatenated to provide a unique identifier and said templates are indexed in accordance with said identifiers.

20. (Original) Apparatus according to claim 19 in which said templates are indexed by said unique identifiers in a look up table.

21. (Original) Apparatus according to claim 13 in which said translation is carried out at runtime of an emulated application program.

22. (Original) Apparatus according to claim 13 in which said templates are provided by software procedure calls.

23. (Original) Apparatus according to claim 13 in which said source code has a 32 bit format and said target code has a 64 bit format.

24. (Original) Apparatus according to claim 13 in which said source code is PA-RISC code and said target code is Itanium^{TM} code.

25. **(Currently Amended)** A computer program product for translating binary code instructions from a source format to a target format for processing by a target processor, comprising a computer-readable medium, further comprising:

a template for use in a binary code translator for translating binary code instructions from a source format to a target format for processing by a target processor, the template comprising:

a template identifier for uniquely associating said template to a source instruction;

a set of target instructions in a target format semantically equivalent to the source instruction;

a set of codes for performing dependency analysis using a counter associated with each instruction to indicate a group number to which the instruction belongs, wherein the counter computes the earliest group to which a given instruction can be assigned such that all producers of inputs and outputs of the instruction have already been assigned to previous groups~~Directed Acyclic Graph~~; and a set of codes for generating dependency analysis code using ~~input and output resources named in the template~~the dependency analysis.

26. (Previously Presented) A computer product according to claim 25, further comprising a set of codes causing a computer to derive the template identifier from a control part of the source instruction.

27. (Previously Presented) A computer product according to claim 26, wherein the template causes a computer to transform a data part of the source instruction into at least one corresponding data part of the set of target instructions.

28. (Previously Presented) A computer product according to claim 27, further comprising a set of codes for causing a computer to bit fill the data part of the source instruction.

29. (Canceled).

30. (Previously Presented) A computer product according to claim 26, wherein the template causes a computer to create the template identifier by concatenating the control part of said source instruction.

31. (Previously Presented) A computer product according to claim 25, wherein the template causing a computer to transform a source instruction having a 32 bit format to a target instruction having a 64 bit format.

32. (Previously Presented) A computer product according to claim 25, wherein the template causes a computer to transform PA-RISC source code into Itanium^{TM} target code.

33. **(Currently Amended)** A computer program product for translating binary code instructions from a source format to a target format for processing by a target processor, comprising:

a computer-readable medium, comprising:

a first set of codes for causing a computer to identify a source instruction;

a second set of codes for causing a computer to select a translation template corresponding to said identified source instruction, said template providing a set of target format instructions semantically equivalent to said identified source instruction;

a third set of codes for causing a computer to translate said identified instruction in accordance with said template;

a fourth set of codes for performing dependency analysis using a counter associated with each instruction to indicate a group number to which the instruction belongs, wherein the counter computes the earliest group to which a given instruction can be assigned such that all producers of inputs and outputs of the instruction have already been assigned to previous groups;

a fifth set of codes for generating dependency analysis code using input ~~and output resources named in the template~~using the dependency analysis; and

a sixth set of codes for causing a computer to output said translated instruction for processing by said target processor.

34-35. (Canceled).